CS501
VOTE: Visual Object Tracing Engine

User Guide

Carrer, Marco: mc120@cornell.edu
Chang, Thomas: hc75@cornell.edu
Lin, Paul: pil3@cornell.edu
Stein, Chris: cas38@cornell.edu

December, 5 1997

User Manual

System Requirements

Currently, VOTE is available only for Windows. It has only been tested on Windows NT 4.0 but may also work under Windows 95 or Windows 3.1. There are no other requirements on the system.

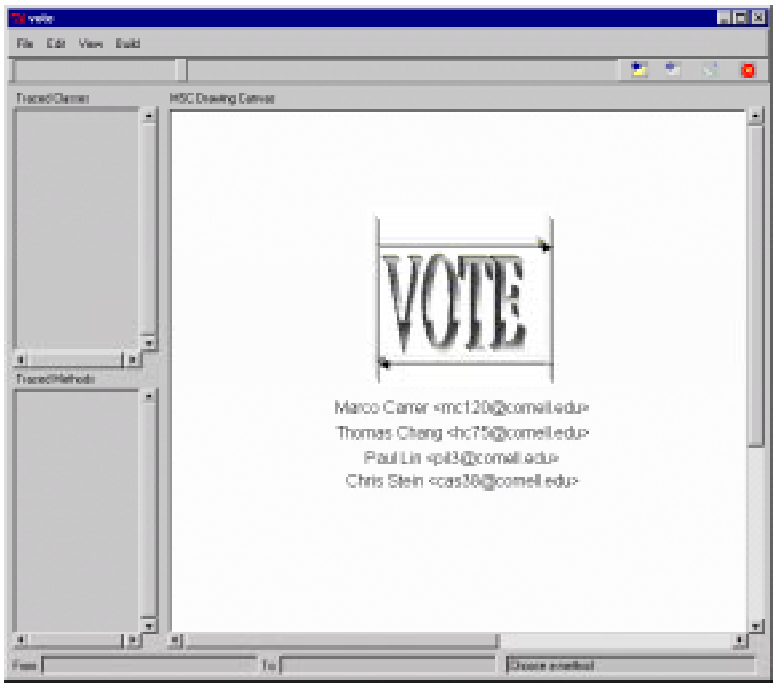
Installing VOTE

First check that you have all the files in the installation package. This includes a self-extracting executable called VOTE_B10.EXE, this document as well as a self extracting installer for Tcl/Tk. Execute VOTE_B10.EXE in a directory where you wish to install VOTE. All the files and the appropriate directory structure will be unzipped into the current directory. Execute the Tcl/Tk installer if you don't have Tcl/Tk. Now you are ready to use VOTE.

Running VOTE

Getting Started

VOTE should be started in the directory that it was installed. This means that there are two subdirectories in the same directory called sources and icons. To start the program, at the command prompt, type in "vote vote.tcl". You will see a window open that looks like this:

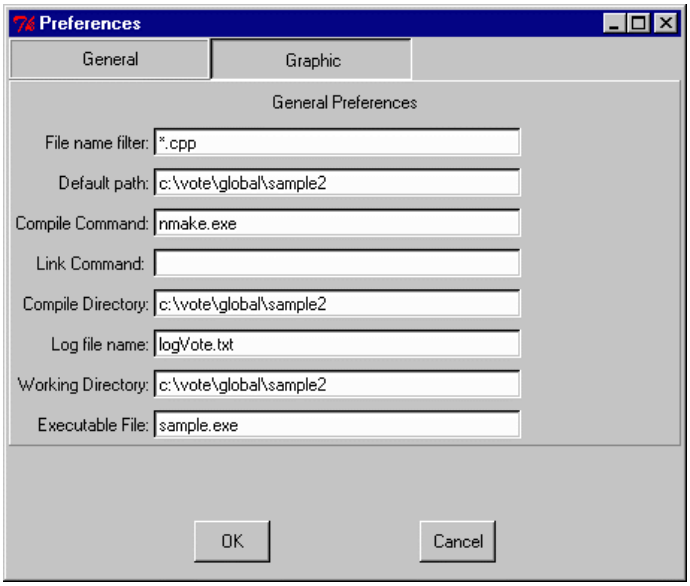


There are numerous options to take at this point. As this tool is geared towards developers who wish to see an analysis of an execution of their program, we're assuming that you have the source files ready to build and execute. For simplicity, we recommend you provide a makefile which is applicable to your chosen compiler to compile the sources. For safety you should try to compile first and see that there is nothing wrong with the source code. Then remove all the binaries and you're ready to run VOTE. For first time users, there aren't any log files generated yet. So you

must generate one by going through each of the steps in the preprocessing stage. For users who have already generated log files in previous executions, you can skip the next two sections.

Setting up Your Preferences

You can modify preferences by clicking *Preferences* in *Menu Option Edit.* You will see a dialog box like this:



There are two categories of preference parameters; General and Graphic.

General Preferences

File name filter: file name extensions for the preprocessor to search for the source files. The default value is "*.cpp"
Default path: directory where the preprocessor starts to search the source file.
Compiler Command: command line input to compile your source code.

Example: gcc -c or g++ -g –c

If you use 'make' to perform the compiling and linking in one step, then type the make application name in this Compiler Command window and skip the next Link Command window.

Example: make.exe _DEBUG=1
Link Command: command line input to link your object files to executables.

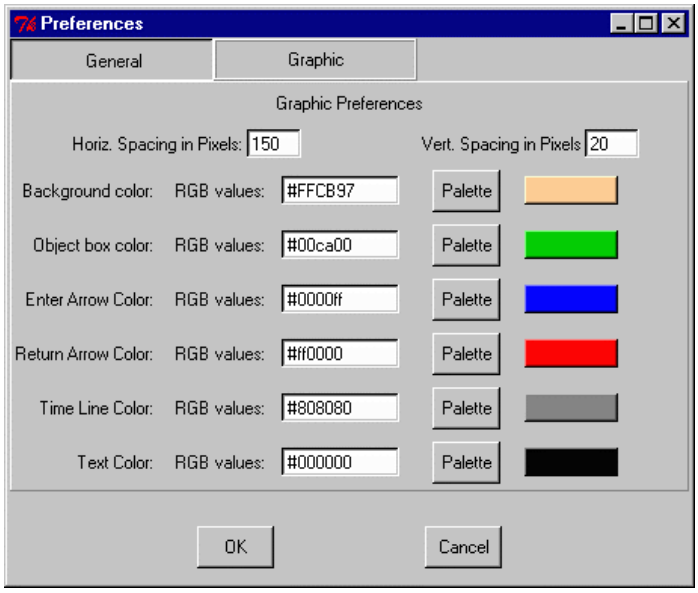
Example: ld –lsocket -lstl

Compiler Directory: directory where the preprocessor will find the makefile and compile.

Log file name: file name of the log file
Working Directory: directory where the executable is generated.
Executable file: file name of the executable in the build process.

Graphic Preferences

Horizontal Spacing in pixels: space between two time lines in pixels
Vertical Spacing in pixels: space between two adjacent event (return/enter) in the time lines, specified also in pixels.

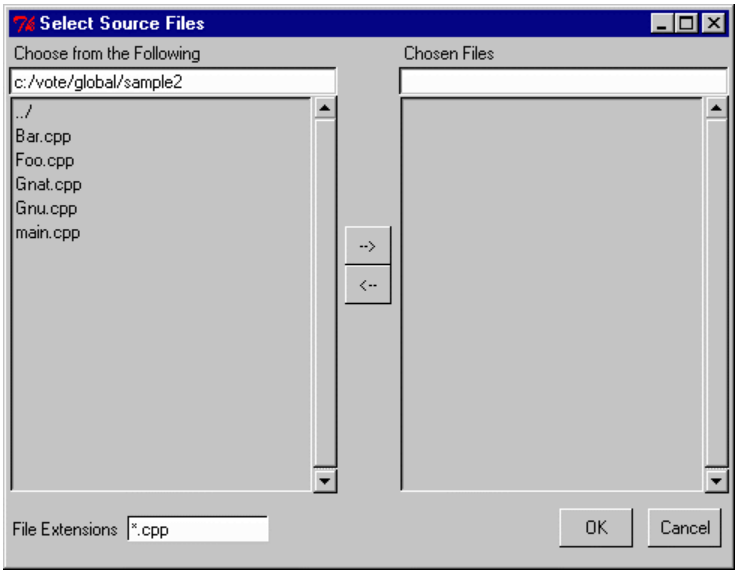


Color Settings: You may specify the color of the specified component shown in the MSC. You can click on the Palette button to choose from a palette or you can type in the RGB values in the appropriate boxes.

The Preprocessing Stage

Selecting Source Files

Go to the *Build* menu button and select the menu item *Select Sources*. A dialog box opens up like this:



You're presented with two list boxes, the one on the left showing you the list of files in the current directory, filtered by the filter given at the bottom of the dialog, and the one on the right showing you all the files you have chosen so far. In the chosen files box, the path is

removed from the file names; but if you wish to know which one it is, you can click on the file name and the entry box at the top of this list box will show you the full path of the file. This list of files persists for the duration of the execution of VOTE.

Modifying Source Files

Once you have selected which files you wish to alter, click on the *Modify Sources* menu item in the *Build* menu button. This will go through each of the files you selected and insert code in there to enable the logging during your program's execution. The status box at the bottom of the main window will indicate that the modification has finished.

Building the Executable

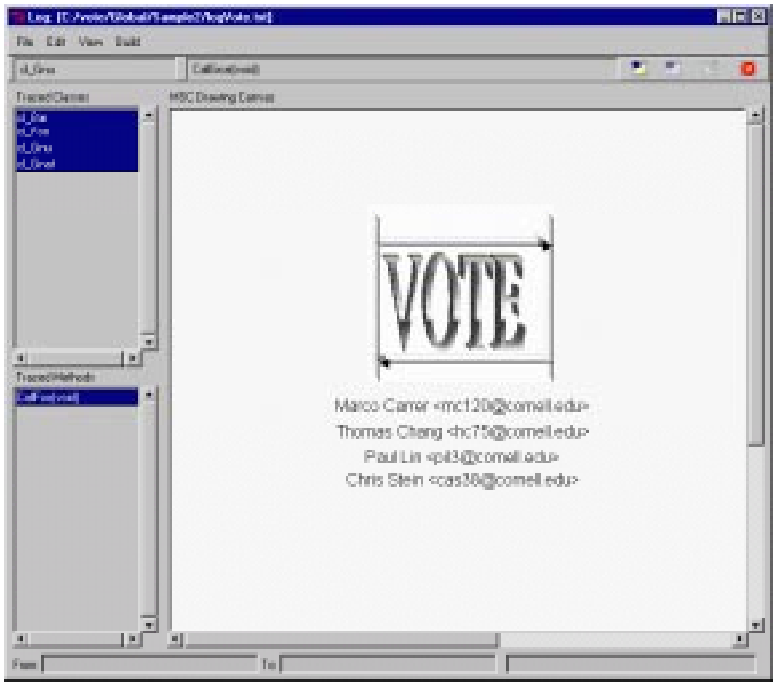
This stage of preprocessing relies on the information given in the preferences. It makes use of the compile command, the link command, and the compile directory. One limitation of VOTE is that if you prefer to build your project in some visual environment you'll need to do the building yourself there and skip this step. VOTE provides only command line compiling functionality, which prevents you from going to another command prompt window to build your executable, if you normally build via command line commands.

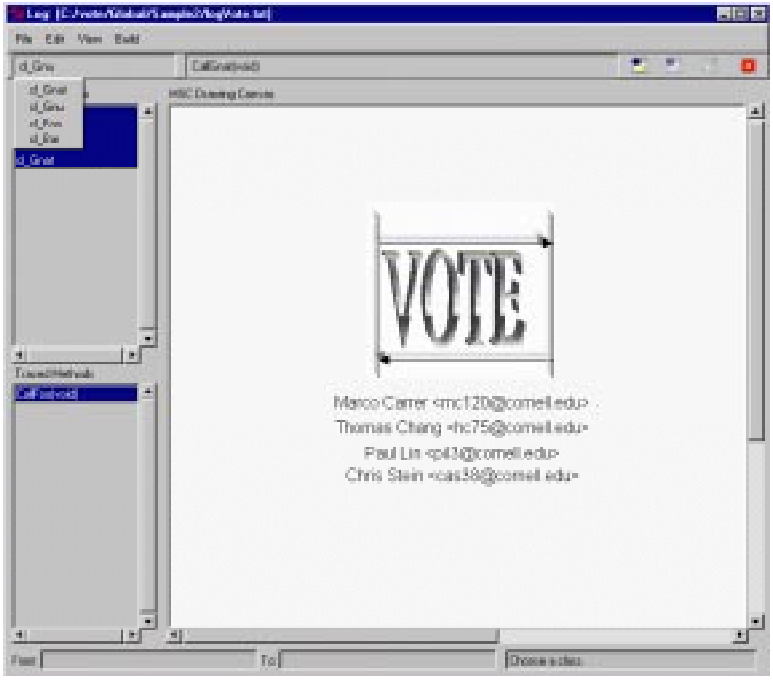
Running Your Executable

The final stage of preprocessing again relies on the information given in the preferences including the working directory and the log file name. It simply invokes your program (which is why the previous step can be skipped if you chose to build using some other visual environment compiler) and generates a log file with the name you chose.

The Log File

Once the log file has been generated, you can load it or view it using the *File* menu button





Loading the Log File

Go to the *Load Log File* option under *File* and choose the log file that you specified in your preferences. Once it is loaded, the two list boxes on the left side of the main window should be updated with class names and method names, as shown in the first figure in the next page.

Similarly the two pull down menus at the top of the window will be updated with similar information. These two pull-down menus let you choose the starting and ending points of drawing the Message Sequence Chart. The left menu indicates the class name and the right one indicates the method name.

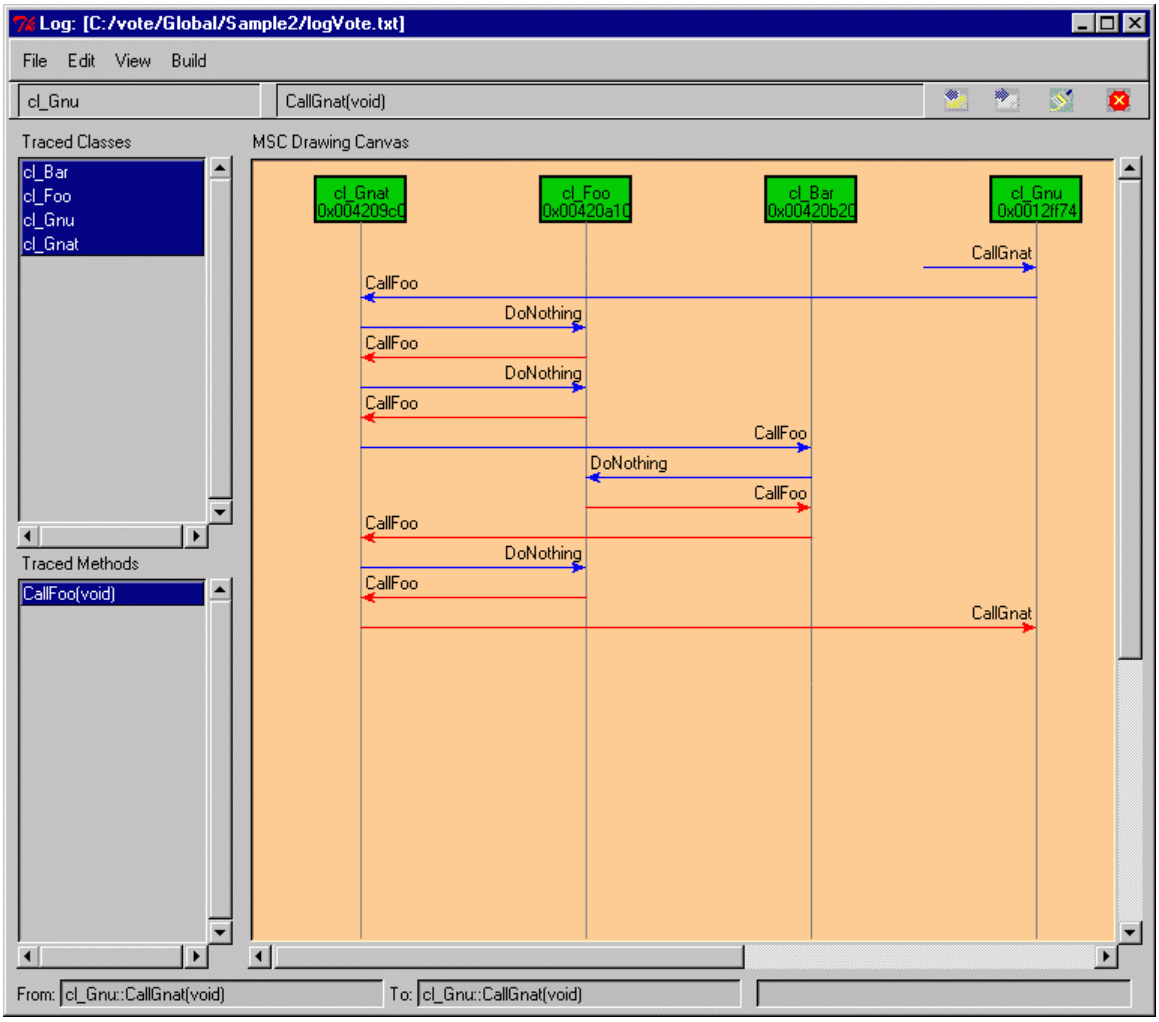
There are four buttons to the right of the pull-down menus. Initially the only button that is enabled is the one indicating the starting point. Clicking on it indicates that you have made your selection. Then it is disabled and the end point button is enabled. Once again you can select the ending point in the same manner. After it has been selected, the draw button is enabled. Finally clicking on the draw button will draw the MSC according to the log file and your selections. The final MSC is shown in the screen shot in the following page.

The two list boxes on the left is your control of selecting whichever time lines you desire to show up on the MSC.

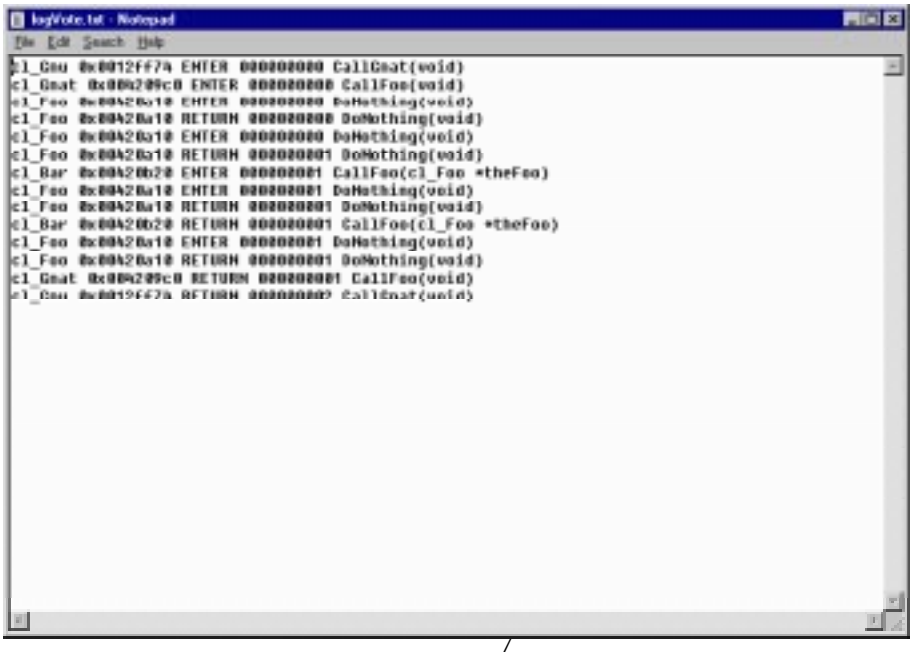
Viewing the Log File

Go to the *View Log File* option under *File* and choose the log file that you specified in your preferences. This will invoke NOTEPAD.EXE and you can view your log file if it interests you.
You can alter it but you're at risk of making it not readable by the MSC drawing engine. This option is designed for expert users who know the format of log files and wish to do some hacking with it to trace errors in their program. For other users it is only an option to satisfy their curiosity of what a log file looks like.

Resulting MSC Diagram:



View the log file:



An Example

In this section we'll guide you through an example that we have set up. The source code can be found in the *Samples/Sample2* under the VOTE directory. Build the executable yourself by typing "nmake" at the command prompt in that directory. We're assuming that you can normally do builds like these, meaning your MSDEV environmental variables should be set already. The executable, called SAMPLE.EXE, simply makes some nested function calls and prints out information while making the calls. After verifying that this works, delete the object files and the executable.

Preprocess the files

1) First launch VOTE by calling "vote vote.tcl" in the *bin* directory under the VOTE directory. Next check using *Edit*/*Preferences* that the preferences are set so that the **Default path**, **Compiler Directory**, and **Working Directory** all point to "VOTE\Samples\Sample2" where VOTE is the VOTE directory. Make sure that you do specify a log file name and that the **Executable File** is Sample.exe. Select OK and now you're ready to modify the files.
2) Go to the *Build* menu button and select the option *Select Sources*. You should see a dialog box with a list of files on the left that comprise the source files inside of *Sample2*. Drag-select all of them and use the right arrow to move them to the box on the right. Clicking on each file gives the full path on the entry box above that list box. Now click OK and exit this dialog box.
3) Go to the *Build* menu button and select the option *Modify Sources*. You should see status information being written in the lower right entry box. Once the box stops outputting new information, the modification is done.
4) Go to the *Build* menu button and select the option *Build*. You're greeted with the hour glass indicting that building is being undergone. Once you have the arrow back, building is done.
5) Go to the *Build* menu button and select the option *Run*. Again you'll see the hour glass and again you should wait until the arrow comes back. Now the preprocessing is done.

Drawing the MSC

6) Go to the *File* menu button and select the *Load Log File* option. In choosing the file, search for the one that was set in the preferences. The default is VoteLog.txt. Then click on OK.
7) You'll see the window being updated with information about the class information of the files that we used. First go to the Class Selection pull down menu right under the *File* menu button and select cl_Gnu.
8) Then go to the Method Selection pull down menu just to the right of the previous pull down menu, and select the only method there; it should be CallGnat(void).
9) That was the starting method that we were selecting in steps 7 and 8. To let VOTE know this, click on the FROM button that is immediately to the right of the Method Selection pull down menu.
10) Go to the Class Selection pull down menu again and this time select the <Any> option. Repeat that for the Method Selection pull down menu and then click on the TO button immediately to the right of the FROM button.
11) The above steps tell VOTE that it should plot all the events that occurred after the entrance into the cl_Gnu::CallGnat function until the end of the log. We're ready to plot now. Simply click on the DRAW button immediately to the right of the TO button and you'll see an MSC representing an execution of the *Sample2* executable.

Trouble Shooting

When you don't see any default values in the Preferences dialog:

Reason: This is an indication that your DEFAULT.INI file is missing from the same directory as the VOTE executable.

Solution: Search in the installation package for the DEFAULT.INI file and copy it to the VOTE directory.

When you get an error message saying that select.tcl, build.tcl, handling.tcl, or CanvasLib.tcl cannot be read:

Reason: This is an indication that one or more of the Tcl script files that came with VOTE, is missing from the ***sources*** directory.

Solution: Identify the missing files in the installation package and copy them to the ***sources*** directory under the VOTE directory.

When you don't see the changes you've made in the Graphic Preferences dialog box after clicking OK:

Reason: You have not clicked on the draw button to redraw the canvas.

Solution: Click on it.

When you don't see the some of the methods of a class show up in the drawing or in the two list boxes:

Reason: The methods you don't see are probably in-line. Even though you may have selected the file containing the implementation of it, the Preprocessing Engine is not capable of detecting its existence.

Solution: Make the method not an in-line function.

When you fail to build or run your application:

Reason: It may be that you could not build or run your application even without using VOTE.

Solution: Please do make sure that you can -build and run your application in its original state. We are not responsible for causing such problems.